# Investigating and Coordinating Safety-critical Feature Interactions in Automotive Systems Using Simulation

Christoph Luckeneder, Michael Rathmair, and Hermann Kaindl
TU Wien, Institute of Computer Technology, Vienna, Austria
{christoph.luckeneder, michael.rathmair, hermann.kaindl}@tuwien.ac.at

## Abstract

*Automotive systems are safety-critical cyber-physical systems. In particular,* undesired feature interaction *can lead to safety-critical behavior. In order to address this problem, we investigate physical feature interaction in this context using simulation (with more than one physical variable). This allows us to visualize both the behavior of features in isolation and their interaction. Our major result is a new insight about feature* coordination*. In such a cyber-physical context, it can be insufficient to coordinate as usual by giving one feature priority over another one. Instead, coordinating based on a physical variable involved in the feature interaction appears to be both necessary and sufficient. In summary, we present our investigation of safety-critical feature interactions and their coordination in automotive systems using simulation, and its results.*

## 1 Introduction

Automotive systems have become software-intensive systems. They are equipped with powerful control units, allowing them to provide increasing numbers of *features*, i.e., characteristics or stakeholder-visible units of behavior. This trend will continue in the foreseeable future, but such features are not independent from each other. This leads to *feature interaction* (FI) [1], i.e., when the interplay of two or more features gives rise to an overall system behavior that is not easily deducible from the individual behaviors of the features involved, and often unexpected.

*Undesired* FI in automotive systems can be safety-critical, since these are cyber-physical systems. While also cyber-physical features are usually implemented in software, investigation of the cyber-physical system is required, and of its interplay with the physical environment.

We performed such investigations by simulating a *hybrid system*, since we connected a physical model with a model of a cyber-physical system [8]. More precisely, we performed *fixed step-size simulation* of corresponding models in Matlab/Simulink, see `http://de.mathworks.com/help/simulink`. Our approach covers visualization, checks of assertions and systematic testing, all based on the same models. Hence, we investigate feature interaction happening in the physical environment, rather than inside the software only. We deal with more than one physical variable (velocity, acceleration and distance).

Visualization during simulation is a novel approach for showing feature interactions. More precisely, we look at the behavior in terms of changes to critical physical variables. We compare this behavior of features running in isolation with the behavior when they are running together. In addition, we combine checking against specifications in this course both through visualization and assertions.

Our major new result from these investigations is an insight about feature *coordination*. First, we tried the approach found in the literature to give one feature as a whole priority over the other (ones) [4, 6]. However, we discovered through our simulations that it can result in undesired behavior. An approach based on a physical variable involved in the FI, in contrast, appears to be both necessary and sufficient for the coordination of such interacting features.

Our running example is based on *Adaptive Cruise Control* (ACC), which is one of the more advanced features ready for penetrating the automotive market [19]. It has already been studied as a Simulink model like in our approach, but with a completely different focus [14]. While ACC is already well understood as a single feature from an engineering viewpoint, we study it from the perspective of a *composite feature*. It includes both Cruise Control (CC), as widely used in cars, and Distance Control (DC). Since ACC per se is well understood already, we can focus in our work on FI simulation and visualization, as well as feature coordination.

The remainder of this paper is organized in the following manner. In order to make this paper self-contained, we briefly provide some background and related work on FI. Then we explain our simulation approach and the models devised for it. Based on that, we present results from our

URI: http://hdl.handle.net/10125/41907
ISBN: 978-0-9981331-0-2

simulations, including a new insight on feature coordination. Also based on our simulation models, we present automated testing for FI using assertions. Finally, we discuss our approach more generally and propose future work.

## 2 Background and Related Work

Developing new features must involve detecting, analyzing and coordinating FI in a scalable manner. Automatically detecting FI is out of the scope of this paper, which focuses on analyzing and coordinating known FI. Even if FI are known, coordinating them poses additional challenges. In particular, the challenge of coordinating FI in a cyber-physical system has not yet attracted enough attention.

While there is an important study of *dependencies* in real-world automotive systems [15], the FI problem in real-world automotive systems has just been touched yet. Static analyses for finding (structural) dependencies along the lines of [15] will be useful for locating potential (behavioral) FI, but for really finding unknown FI, much more work will have to be done.

An approach for detecting FI automatically through *model checking* can be found in [7]. It finds conflicts on one variable (speed as in our case), but does not involve any physical model, so that any effect on a distance to another vehicle cannot be included.

In the context of software(-only) features, FI detection was discussed as a potential application of a software monitoring approach in [10].

Previous work on modeling ACC in Simulink [14] did not focus on features or their interaction in a physical model. In particular, the simulation did not cover the dependency of speed and distance. Cruise control and distance control were not modeled independently, as the cruise control module also made coordination decisions. Essentially, this work investigated controllers for implementing ACC.

To our best knowledge, there has not yet been much work published on coordinating feature interaction. Jackson and Zave [6] presented early and seminal work on coordinating FI in the telecommunications domain. In essence, this approach avoids undesired FIs through central control, which implements serialization for disabling a feature in favor of another one.

Ertl et al. [4] used the Mediator software pattern in order to reduce the coupling among feature implementations in automotive software. The coordinator uses given compiled knowledge on FI to give one feature priority over others.

Bocovich and Atlee [2] addressed FI resulting from conflict of features accessing the same software variable(s) at the same point in time. Since the granularity of features may be relevant, resolution for each software variable under conflict was proposed and implemented in the automotive domain. This work did not investigate the physical effects from an FI of a software variable, and it did not include independent physical variables outside the control of the software.

Wilson et al. [18] also addressed avoiding undesired FI only, in the domain of building automation. They proposed a layered architecture of a central software coordinator, for locking resources in case of undesired FI. In effect, also this approach implements disabling features.

Prehofer [11] proposed an approach based on statechart diagrams for modeling features. Through integration of such statechart diagrams, non-functional feature interactions can be avoided.

In contrast, Zhang [20] and Lai et al. [9] proposed greedy algorithms for optimization of desired feature interaction, reminiscent of hill-climbing. Wagner et al. [16] additionally included soft and hard constraints, and dynamic adjustment of influence in the utility function for handling feature interactions.

## 3 Our Modeling and Simulation Approach

Let us present first our physical model and models of the cyber-physical features CC and DC, as well as of their coordination. Then we explain the environment for the simulations of these models.

### 3.1 Physical Model

Our physical model for the simulations includes the physical variables distance, speed and acceleration, as well as their dependencies (positive or negative). However, we consider any effect from masses, e.g., outside the scope of these simulation models. Figure 1 illustrates this model as a physical chain of these values. The distance of the vehicle A under consideration to a Vehicle B in front of it depends on speed B (in addition to speed A, of course), which we consider an independent variable, since it is out of control of vehicle A (this dependency is given through a model of the driving behavior of vehicle B). Speed A depends on the acceleration of the vehicle under consideration (through a differential equation). According to [13], in this model only longitudinal motion can be simulated. Simulating the steering angle for overtaking maneuvers, for example, would require a more elaborate model, but this is out of the scope of our investigations.

Vehicle A is the cyber-physical system interacting with this physical chain through requests on a physical quantity (visualized through a green circle). In the context of this paper, we only allow speed requests. Of course, this is a simplification, since actually a certain acceleration would have to be requested that is intended to lead to a requested speed.

Feature interactions may result directly from conflicting

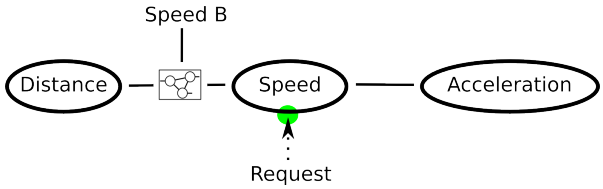



Figure 1: Physical chain

requests to a single variable (as studied for speed before in the context of model checking in [7]). This is sufficient for detecting that there is a feature interaction, but not for investigating its influence on the physical system. We studied this influence on the distance to another vehicle, which is important for simulating the features CC and DC together.

We *validated* the physical part of the simulation model needed for that against the related behavior in the real world. According to [17], we performed desk checking, bottom-up testing and reviews for this purpose.

### 3.2 The Cyber-physical Features

Now let us explain the models of our cyber-physical features and how these influence the physical system. Of course, the physical system needs to be in balance at any time, in order to be realistic. Therefore, only a single request for a single physical variable can be handled at any point in time. This is fulfilled in the approach taken in this paper, since we only allow speed requests, and only one at a time.

Figure 2 illustrates that the features DC and CC can only request a certain speed after a coordinator C resolves a potential conflict. It needs to handle requests coming from both DC and CC. Whenever each of them runs alone, the coordination is trivial, of course. Indirectly in the physical system, such a speed request influences the dependent distance to vehicle B as well.

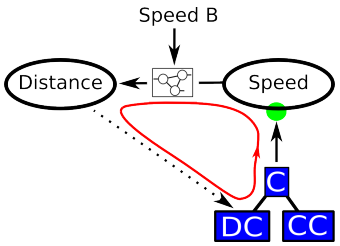



Figure 2: Part of physical chain together with feature model

The dotted arrow in the figure indicates that a measured distance value (in reality provided by a sensor) is fed back to DC, so that DC can keep control on a target distance. Obviously, this leads to a control loop, illustrated in red in the figure. More precisely, a closed-loop controller implements DC. In contrast to an engineered solution for a real vehicle, we did not care about the quality of such a controller. Any speed request from CC, which may run concurrently to DC, may influence this system as well.

For facilitating both visualization and automated tests, specifications of the features are needed. Both CC and DC need an explicit signal each, whether they are activated or not (in reality be the driver, in a simulation run by its particular setting). The only other input of CC is a given target speed. We specify for CC that the speed may only deviate from this target speed by at most $\pm 5\%$. The input of DC comprises a given target distance, the current distance to vehicle B (in reality measured by a sensor), and the speed of vehicle B (in reality an estimate). For DC, we specify that the distance between vehicles A and B may never be smaller than $90\%$ of the given target distance. In addition, the variant of DC investigated here both tries to approach towards a vehicle in front of it until it reaches the target distance, and to keep the target distance. For the composite feature ACC, we specify the upper speed limit given by CC and the lower distance limit given by DC.

Technically, these features are modeled as blocks where state machines implement their behavior, together with Discrete Time Blocks from the Simulink library.

### 3.3 Feature Coordination

Also feature coordination needs to be modeled for our investigations. Let us include here a coordinator according to the Mediator software pattern as introduced in [4]. In the following, we call it Mediator-Coordinator (Med.-Co).

For its inclusion into our cyber-physical model, the feature models need to be technically extended, so that they can signal their Boolean requests to the coordinator. More precisely, two extensions are required. The first one determines, whether a feature intends to issue a request in the given state of the cyber-physical system. The second extension actually puts the value calculated for its speed request to the corresponding variable, if the coordinator releases this feature.

### 3.4 Simulation Environment

As the simulation environment for our models as explained above, we primarily use the Matlab/Simulink modeling and simulation software. Matlab/Simulink supports block-oriented graphical representation of simulation models. The simulation methodology is defined by the TDF (timed data flow) model of computation, in combination with a differential equation solver.

While this environment provides great support for modeling and simulation per se, we could not find how to di-

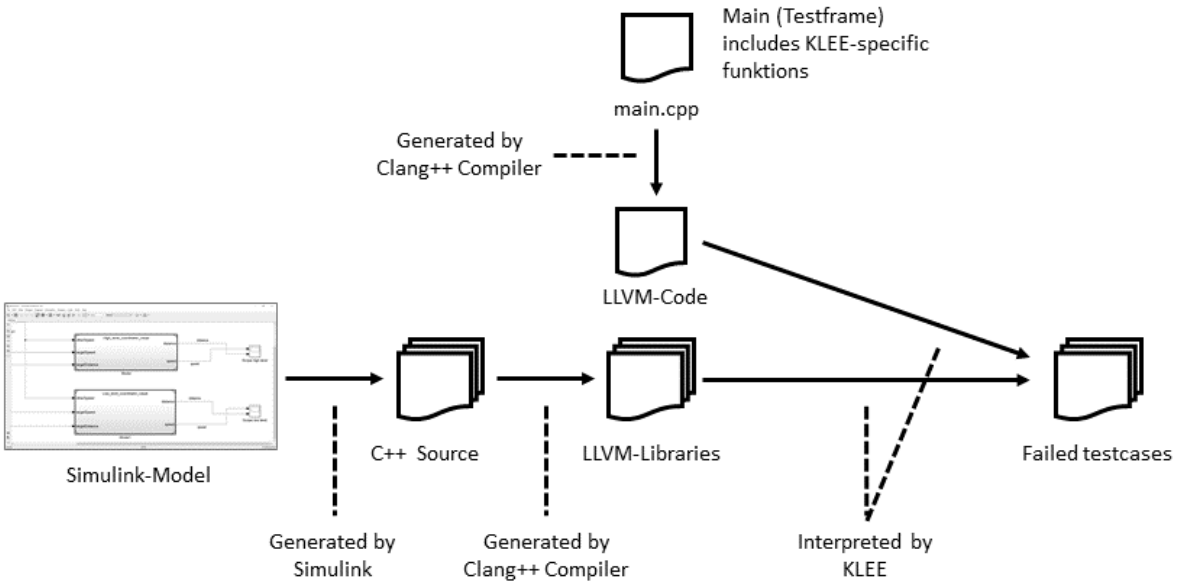



Figure 3: Data-flow and tool chain from Simulink-Model to failed test cases

rectly generate and run an automated test suite (for several different models of the driving behavior of vehicle B). So, we additionally use KLEE, a symbolic virtual machine for interpreting LLVM code, see `http://llvm.org/pubs/2008-12-OSDI-KLEE.pdf`. KLEE is capable of automatically generating test data that achieve high coverage on a diverse set of complex and environmentally-intensive programs.

Still, we wanted to use the same models as implemented in Matlab/Simulink. This requires integration of these tools for a tool chain as illustrated in Figure 3. C++ source code is generated from the Simulink-Model and compiled into LLVM-Libraries. More precisely, the Simulink-Model consists of two parts, one for the features and another one for the physical model. So, at least two C++ files are generated and compiled.

An additional manually provided main.cpp file serves as test harness. It uses both functionality from these generated LLVM-Libraries and KLEE-specific constructs.

The automated tests check for violations of assertions, which correspond to the feature specifications given above. KLEE supports specifying such assertions (through klee_assert). Whenever in the course of a test run a given condition is not fulfilled, KLEE stops this run and documents this failed test. Then KLEE resumes with the given test suite.

## 4 Simulation and Visualization of Results

In Matlab/Simulink, we simulated the models described above and visualized the results. In the course of the simulation process, the physical chain is calculated sequentially.

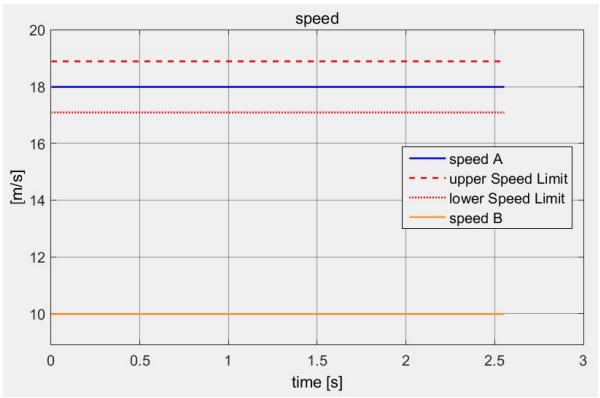
We simulated the cases when CC or DC, respectively, are active alone, and cases where both CC and DC were assumed to have been activated at the same time by the driver of vehicle A. In all cases, we assumed that activated features are not deactivated in the course of the simulation runs, even not in a situation where CC leads to a rear-end collision accident. The target speed for vehicle A was uniformly set to 18m/s, and the target distance to 10m.

For the driver behavior of vehicle B, we provided a simple model about its speed, which is first smaller than that of vehicle A, but suddenly increases after vehicle A has reached vehicle B. This model was used for all the simulation runs reported below, and is depicted in the figures as speed B.

Based on all that, the features CC and DC issued their respective speed requests, which were coordinated according to different strategies as explained below. The calculated speed A is plotted over time in the figures below, depicted as a step-function. The upper and lower speed limits according to the specifications are plotted as well in some of the figures, depicted as broken and dotted lines. In addition, speed B and the resulting distance are plotted over time, depicted as continuous lines (used here for physical variables).

### 4.1 Behavior with CC only

Figure 4 shows the simulation behavior when feature CC is active only, i.e., feature DC is inactive. With regard to speed, everything looks good at a first glance, since the specification of CC is not violated. However, the distance



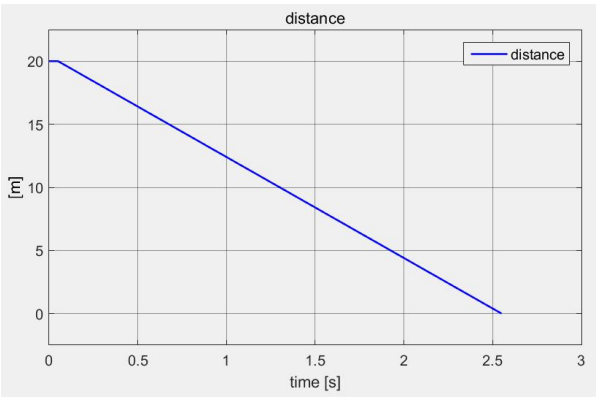




Figure 4: Feature CC active only

becomes 0 due to the higher speed of vehicle A than that of vehicle B, in effect resulting in a rear-end collision accident. This behavior is consistent with the real-world behavior of a vehicle running CC in such a situation, without any intervention. At this point, the simulation stops, since modeling such an accident per se is out of the scope of this work.

### 4.2 Behavior with DC only

Figure 5 shows the simulation behavior when feature DC is active only, i.e., feature CC is inactive. Since vehicle A is faster than vehicle B, there is an approach first, up to the given target distance. Once vehicle B speeds up, vehicle A increases its speed as well, in order to keep the target distance. It is easy to see that the specified lower distance limit is not violated by using feature DC (at least not in this simulation run).

Technically, the immediate acceleration of each vehicle relates to ignoring masses, of course. The control problem visible on the right side of the bottom diagram (distance) of the figure is caused by using a simple proportional controller.

### 4.3 Behavior with Mediator-based Coordinator Med.-Co

Figure 6 shows the simulation behavior when both features CC and DC are active, and coordinated by a Mediator-based coordinator as proposed in [4]. As explained also above, it gives one feature as a whole priority over the other. DC's request has priority whenever the target distance is reached (or the distance is even smaller than the target distance), since rear-end collision accidents are to be avoided. Otherwise, CC's request has priority, so that the target speed is to be kept. In effect, this coordinator makes sure that only one of these features' requests for speed can influence the physical system at any point in time.

The resulting behavior shown in Figure 6 is much more complex than what is shown above for these features in isolation. As long as the target distance is not reached, CC lets vehicle A approach vehicle B, driving with a speed within the specified limits. Once the target distance is reached, however, DC takes over and reduces speed A (roughly) to speed B, in order to keep the distance. While this behavior obviously violates the specification of CC with regard to the lower speed limit, the specification of the composite feature ACC is only slightly violated, by getting a bit too close after the approach due to switching when the target distance is already reached and the simple controller used.

A violation of the specified upper speed limit, both of CC alone and of ACC, occurred, however, between $t = 6$ and $t = 7$. The basic reason is given by the fact that DC alone had full control as granted by this (kind of) coordinator, and DC does not take the target speed of CC into account. DC actually stays active in this approach, as long as it can keep the target distance (depending also on its controller). Only after the distance became a bit larger after $t = 7$, this coordinator assigned the control back to CC, which reduced the speed again.

Such a switch of control from DC to CC actually happened already before, between $t = 5$ and $t = 6$. It was more or less immediately followed by yet another switch from CC to DC. This rapid switching appears to be oscillating behavior.

### 4.4 Behavior with Mediator-based Coordinator and Hysteresis

The well-known means to address oscillating behavior is to employ a *hysteresis*. It was implemented in such a way that a small interval around the target distance was used by the coordinator for switching from DC to CC and vice versa. Figure 7 shows the simulation behavior when both features CC and DC are active, and coordinated by the

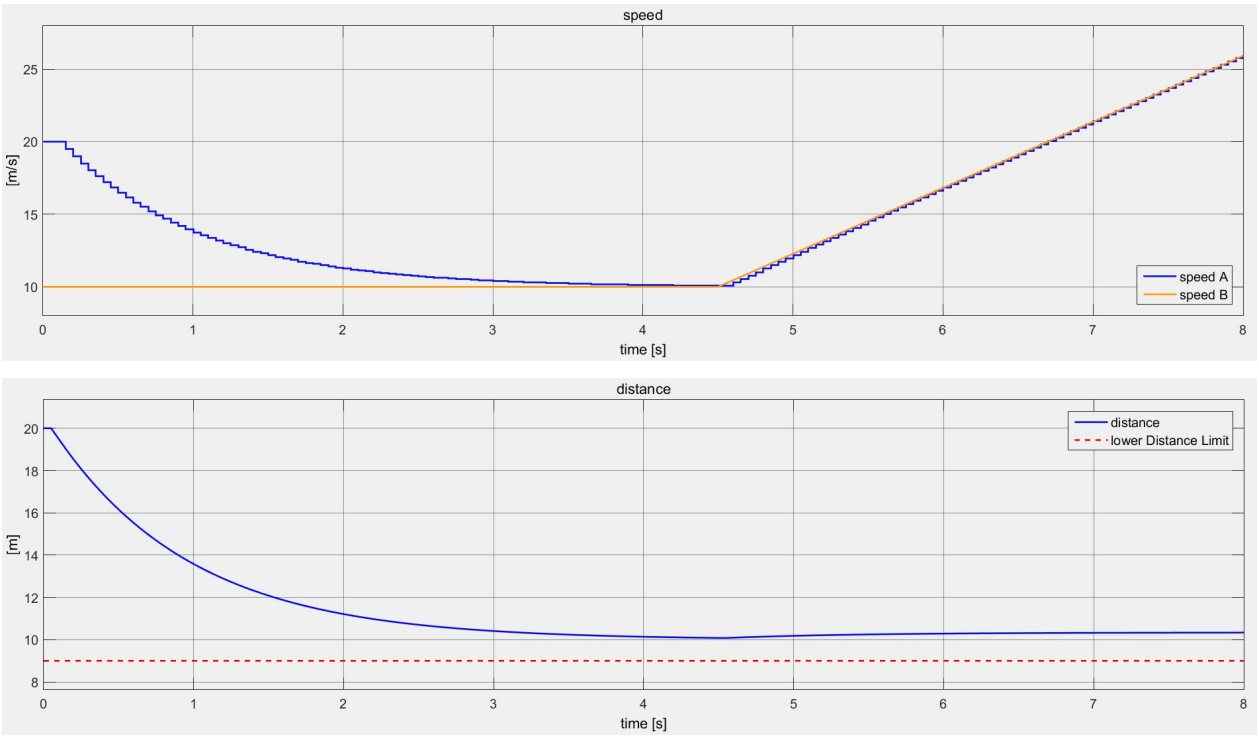



Figure 5: Feature DC active only

same Mediator-based coordinator, in this run with a hysteresis, however. In fact, the apparent oscillating behavior above disappeared (also providing evidence that it actually was oscillating behavior).

However, the violation of the specified upper speed limit occurred again after $t = 6$. In this simulation, the speed exceeded its limit to a larger extent, and the switch back to CC did not even happen. It seems as though speed A would increase depending on speed B.

In effect, the hysteresis made the real problem of this way of coordinating CC and DC even worse. Obviously, the usual approach to give one of the coordinated features simply priority over the other one(s) does not work for this kind of FI.

### 4.5 Behavior with Engineered Coordination

Fortunately, ACC has already been engineered for its use in practice, where it apparently has been devised as a single feature optimized for smooth control of real vehicles. In essence, the approach described in [19] takes the minimum of the acceleration values that cruise control and distance control request, and ACC requests this minimum acceleration.

Hence, an alternative to the Mediator-based coordinator can be informed by this approach. Since the features in our approach request speed values, however, we let our coordinator take the minimum of speed requests. We call it Engineered-Coordinator (Eng.-Co). Technically, both CC and DC provide their separate speed requests in variables as before, from where the alternative coordinator takes them, determines their minimum value and simply assigns it to the variable containing the speed request for the physical model. Such a coordination could also be specified using the resolution formalism of [2].

When considering our feature specifications given above, this approach makes perfect sense. Assuming that both CC and DC do not violate their specifications for the upper speed limit and the lower distance limit, respectively, then our specification of the composite feature ACC cannot be violated by using this coordinator.

Figure 8 shows the simulation behavior when both features CC and DC are active, and coordinated by the Engineered-Coordinator. In fact, there are neither violations of the ACC specification nor any oscillating behaviors. In addition, the approach to vehicle B happens much more smoothly than shown above. Hence, this simulation run provides some empirical evidence that this kind of coordination is sufficient for this kind of FI.

For a further illustration of the FI between CC and DC, we included an additional plot with their respective speed requests in (the middle of) Figure 8. Since both CC and DC are active, these requested values were not directly fed

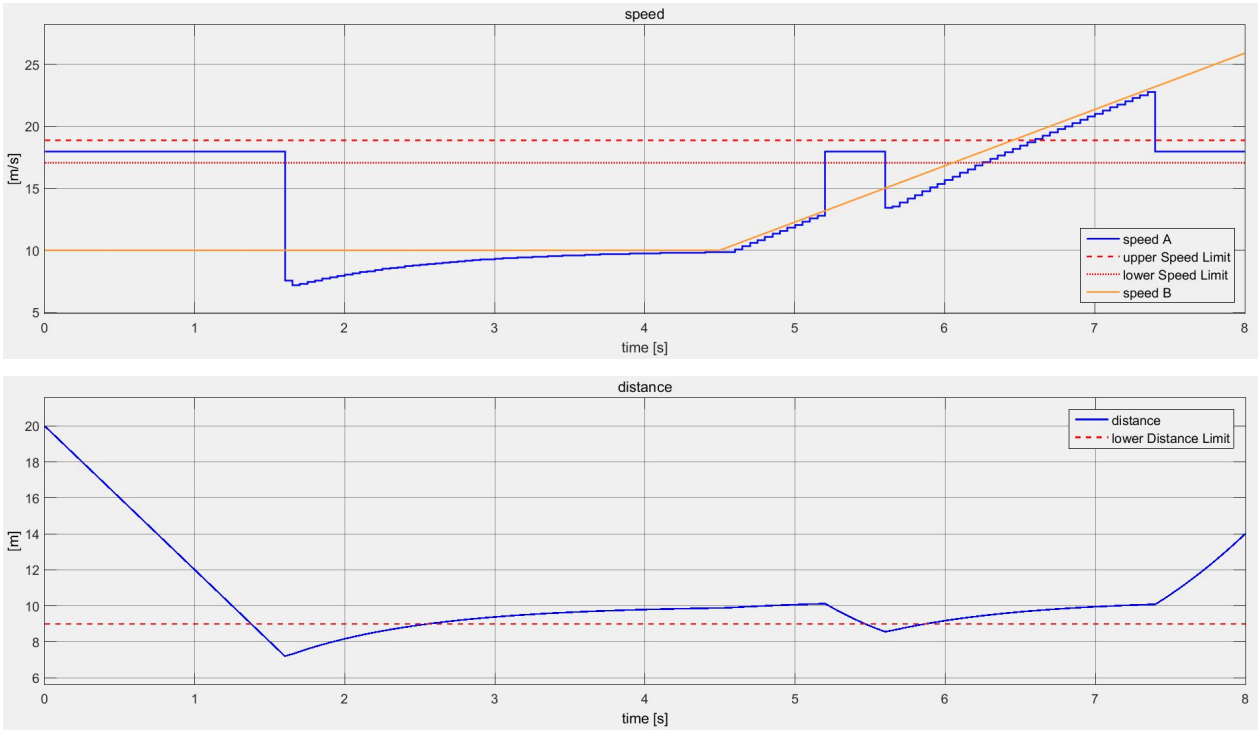



Figure 6: Features CC and DC active with Mediator-based coordinator Med.-Co

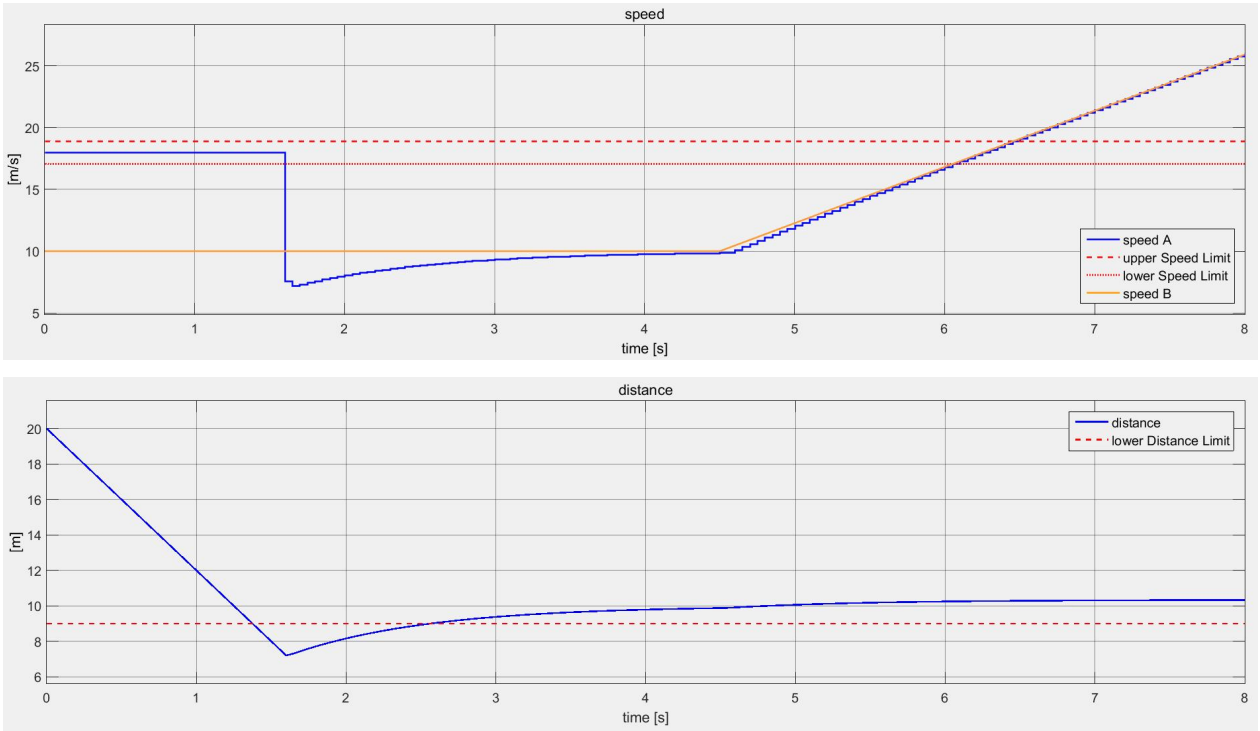



Figure 7: Features CC and DC active with Mediator-based coordinator Med.-Co and hysteresis

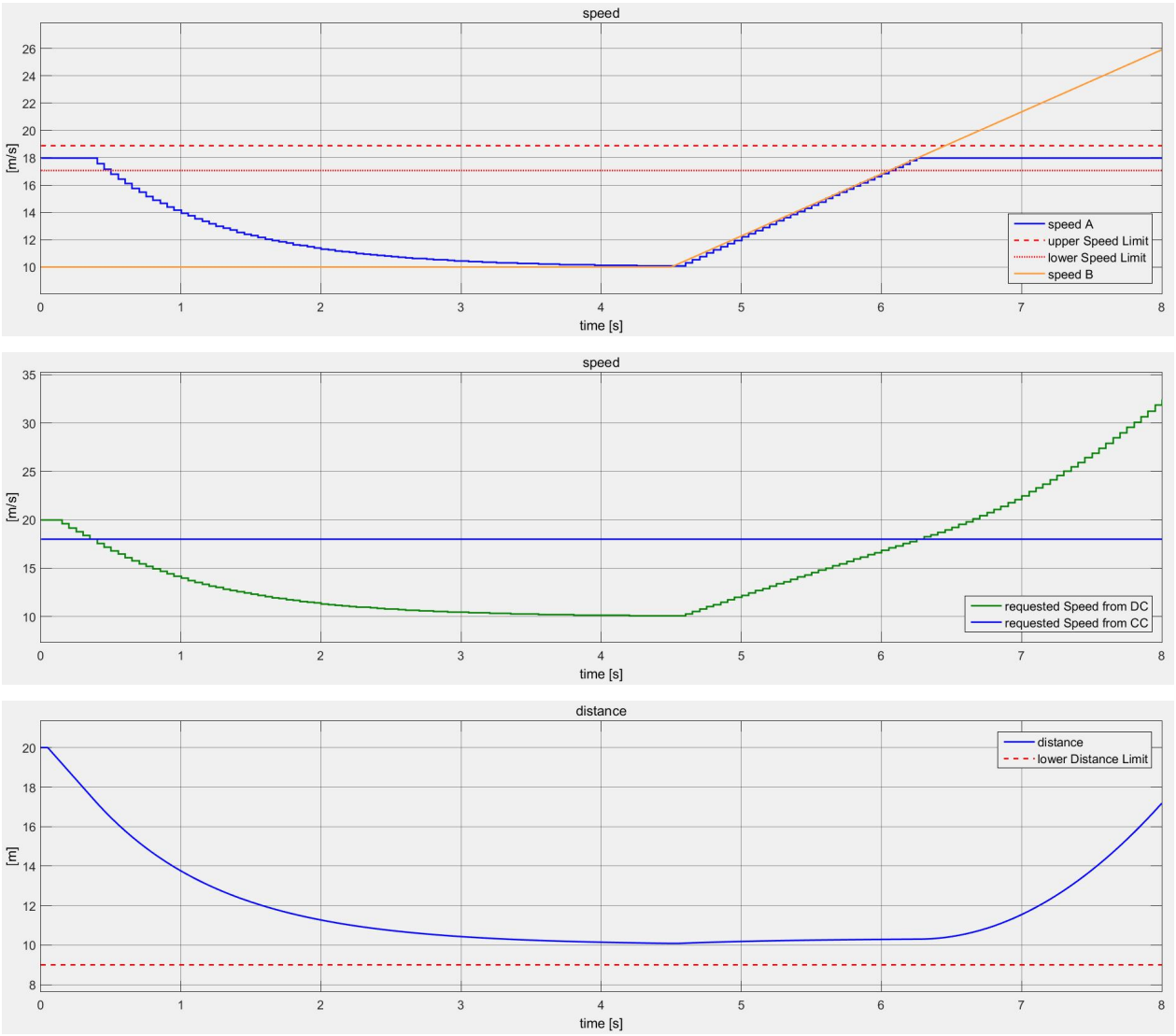



Figure 8: Features CC and DC active with Engineered-Coordinator Eng.-Co

into the physical model, but only served as input for Eng.-Co. Interestingly, these speed requests are identical only at two points in time, where these lines intersect. Otherwise, they are different, which supports the usual view that these features interact.

Eng.-Co directly gets these values as its input, and uses their minimum for coordination. In terms of information flow, this amounts to much more information than the Boolean one that the Mediator-based coordinator Med.-Co gets, and the Boolean decision it takes for coordination. For coordinating such an FI of a cyber-physical system, this makes the difference. In fact, the physical variable at the core of the conflict (in our case speed) is real-valued.

## 5 Automated Tests for FI using Assertions

While these simulations showed interesting behavior and helped to gain a new insight, they were all run with the very same driving behavior of vehicle B. In order to see whether these results hold more generally, we also performed automated tests for FI (with the same simulation models), see also [12]. These tests for *verification* checked for violations of assertions based on the lower and upper limits of the feature specifications. As explained above, these tests ran with the tool KLEE (together with the constraint solver STP) instead of the Simulink simulation software.

Especially for such safety-critical features, a sufficient test coverage is important. Unfortunately, KLEE cannot

generate test data fully-automatically when working together with an external solver such as STP. This problem is supposedly related to the data-flow specifications used in the Matlab models. Still, KLEE generated the combination of all the values as defined for our own test harness, implemented in main.cpp.

According to the *classification tree method* [3], we determined the inputs for the features, as well as the initial speed B and acceleration B. For each of these, we defined equivalence classes for their respective values. For each equivalence class, a single representative value was selected for the test case. In addition, according to *boundary value analysis* [3], we took the boundary values of the target speed and the target distance.

While the simulation runs reported above all used the same driving behavior of vehicle B, we varied several different scenarios in the automated tests. Speed B was restricted, however, to the interval from 0 to 50m/s (the same interval as the one for the target speed of vehicle A). Six different initial values were used for speed B. Acceleration B was assigned three different values in the range $\pm 1 m/s$, and it was only possible to be changed after one second. The duration of each cycle in the test runs was 10s (or less). A cycle was shorter than 10s, if an assertion was violated, or if the value of speed B became out of its specified interval.

KLEE ran the tests for all the given combinations. Hence, the theoretically possible number of paths executed amounts to a maximum of $3,188,646$ ($= 3 * 3 * 6 * 3^{10}$).

Table 1 summarizes the test results (using KLEE and the test harness as specified above). The most important result is that Eng.-Co did not violate any single assertion, while Med.-Co did that in many cases. While this is, of course, no correctness proof of Eng.-Co, it provides some empirical evidendence for Eng.-Co being suffiecent for coordinating CC and DC.

| | Med.-Co | Eng.-Co |
|---|---|---|
| Theoretical number of paths | 3,188,646 | 3,188,646 |
| Number of investigated paths | 966,774 | 2,386,224 |
| Violations of distance assertion | 16,952 | 0 |
| Violations of speed assertion | 79 | 0 |

Table 1: Results of automated tests

In more detail, the number of violations of the distance assertion by Med.-Co is much higher than the number of violations of the speed assertion. The difference in the numbers of investigated paths can be explained as follows. Eng.-Co could not terminate any single path because of a violated assertion, but only if the value of speed B became out of its specified interval.

Since these are safety-critical features, ISO 26262 [5] has to be applied in practice, which requires a specified test coverage. For our more theoretical investigations, it was out of scope to show whether these tests achieve the required coverage or not. Unfortunately, KLEE would not provide support for that, hence yet another tool (e.g., gcov from the GNU Compiler Collection (GCC)) would be required.

## 6 Discussion and Future Work

Originally, the concern of FI coordination was to make sure that the behavior of interacting features does not result in undesired overall behavior, e.g., by giving one feature in telecommunications priority over another one. In our case, in contrast, an FI coordinator of a composite feature (in our case ACC) needs to create the desired behavior of this composite feature from the conflicting requests of its component features (in our case CC and DC). Does this relate to the concept of desired FI studied in previous work? Feature coordination may also be viewed as creating desired behavior from FI.

Clearly, the two coordination approaches investigated in our paper differ in the amount of information exchanged between the respective coordinator and the features it deals with. Is there a dependency between the amount of information to be exchanged in a coordination approach and its capability to create desired or at least to avoid undesired behavior, respectively?

These questions, among others, indicate lack of a theory of feature coordination, whose creation will be important in future work.

## 7 Conclusion

In this paper, we present an investigation of FI in a cyber-physical (automotive) system using simulation, for an improved understanding of such FI in contrast to FI in software alone. Also for a cyber-physical system, certain FI detection is possible through conflicting requests on the same variable within the software. However, for studying the interacting behavior in a physical system, we employed simulation of the physical system that the cyber-physical system interacts with. This simulation involved an additional physical variable that depends on the behavior of an external agent — the distance to a vehicle in front.

These simulations helped us to gain a new insight on coordination of FI. The traditional approach to give priority to one feature and to inhibit the other(s) is insufficient for coordinating (certain) FI in a cyber-physical system. Its Boolean input and Boolean decision can lead to undesired effects. In contrast, coordination based on the real-valued physical variable under conflict was both necessary and sufficient in our case. We conjecture that more information has to be involved for coordinating FI in a cyber-physical system than in software alone. Hence, FI in cyber-physical systems is

more intricate than FI in software alone, especially its coordination.

## Acknowledgment

Part of this research has been carried out in the FeatureOpt project (No. 849928), funded by the Austrian BMVIT (represented by the Austrian FFG).

We thank Sven Apel for his hint to include the limits given by the specification in our plots.

## References

[1] S. Apel, D. Batory, C. Kästner, and G. Saake. *Feature-Oriented Software Product Lines: Concepts and Implementation*. Springer, 2013.

[2] C. Bocovich and J. M. Atlee. Variable-specific resolutions for feature interactions. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 553–563. ACM, 2014.

[3] B. Broekman and E. Notenboom. *Testing embedded software*. Pearson Education, 2003.

[4] D. Ertl, S. Dominka, and H. Kaindl. Using a mediator to handle undesired feature interaction of automated driving. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 4555–4560, Oct 2013.

[5] ISO 26262. ISO 26262, Road vehicles – Functional safety, Nov. 2011.

[6] M. Jackson and P. Zave. Distributed feature composition: A virtual architecture for telecommunications services. *IEEE Transactions on Software Engineering (TSE)*, 24(10):831–847, 1998.

[7] A. L. Juarez-Dominguez, N. A. Day, and J. J. Joyce. Modelling feature interactions in the automotive domain. In *Proceedings of the 2008 International Workshop on Models in Software Engineering*, MiSE '08, pages 45–50, New York, NY, USA, 2008. ACM.

[8] H. Klee and R. Allen. *Simulation of dynamic systems with MATLAB and Simulink*. CRC Press, 2011.

[9] H. Lai, Y. Tang, H. Luo, and Y. Pan. Greedy feature selection for ranking. In *Computer Supported Cooperative Work in Design (CSCWD), 2011 15th International Conference on*, pages 42–46. IEEE, 2011.

[10] P. Lengauer, V. Bitto, F. Angerer, P. Grünbacher, and H. Mössenböck. Where has all my memory gone?: Determining memory characteristics of product variants using virtual-machine-level monitoring. In *Proceedings of the Eighth International Workshop on Variability Modelling of Software-Intensive Systems*, VaMoS '14, pages 13:1–13:8, New York, NY, USA, 2013. ACM.

[11] C. Prehofer. An adaptive control model for non-functional feature interactions. In *Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on*, pages 501–507. IEEE, 2011.

[12] R. G. Sargent. Verification and validation of simulation models. In *Simulation Conference (WSC), Proceedings of the 2010 Winter*, pages 166–183, Dec 2010.

[13] D. Schramm, R. Bardini, and M. Hiller. *Vehicle Dynamics*. Springer, 2014.

[14] V. Sivaji and M. Sailaja. Adaptive cruise control systems for vehicle modeling using stop and go manoeuvres. *International Journal of Engineering Research and Applications (IJERA), ISSN*, pages 2248–9622, 2013.

[15] A. Vogelsang and S. Fuhrmann. Why feature dependencies challenge the requirements engineering of automotive systems: An empirical study. In *Requirements Engineering Conference (RE), 2013 21st IEEE International*, pages 267–272. IEEE, 2013.

[16] D. Wagner, H. Kaindl, S. Dominka, and M. Dübner. Optimization of feature interactions for automotive combustion engines. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 1401–1406. ACM, 2016.

[17] R. B. Whitner and O. Balci. Guidelines for selecting and using simulation model verification techniques. In *Proceedings of the 21st conference on Winter simulation*, pages 559–568. ACM, 1989.

[18] M. Wilson, E. H. Magill, and M. Kolberg. An online approach for the service interaction problem in home automation. In *Consumer Communications and Networking Conference, 2005. CCNC. 2005 Second IEEE*, pages 251–256. IEEE, 2005.

[19] H. Winner and M. Schopper. Adaptive cruise control. In *Handbuch Fahrerassistenzsysteme*, pages 851–891. Springer, 2015.

[20] T. Zhang. Adaptive forward-backward greedy algorithm for learning sparse representations. *Information Theory, IEEE Transactions on*, 57(7):4689–4708, 2011.